(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 507 571 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **92302872.4**

(51) Int. Cl.[5] : **G06F 5/06**

(22) Date of filing : **01.04.92**

(30) Priority : **05.04.91 JP 73146/91**

(43) Date of publication of application :
07.10.92 Bulletin 92/41

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **FUJITSU LIMITED**
1015, Kamikodanaka Nakahara-ku
Kawasaki-shi Kanagawa 211 (JP)

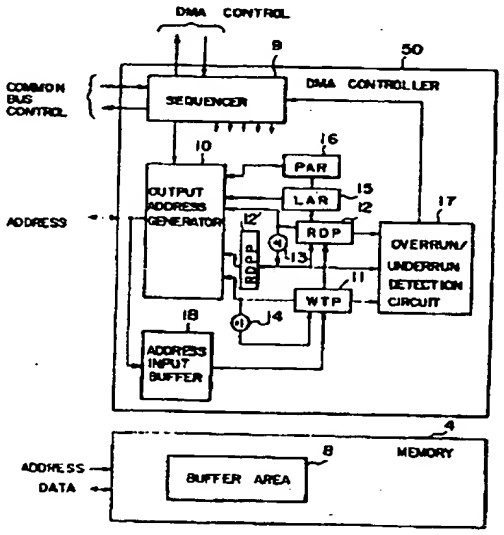
(72) Inventor : **Shimizu, Toshiyuki, c/o Fujitsu Limited**
1015, Kamikodanaka, Nakahara-ku
Kawasaki-shi, Kanagawa 211 (JP)
Inventor : **Horie, Takeshi, c/o Fujitsu Limited**
1015, Kamikodanaka, Nakahara-ku
Kawasaki-shi, Kanagawa 211 (JP)
Inventor : **Ishihata, Hiroaki, c/o Fujitsu Limited**
1015, Kamikodanaka, Nakahara-ku
Kawasaki-shi, Kanagawa 211 (JP)

(74) Representative : **Billington, Lawrence Emlyn et al**
Haseltine Lake & Co Hazlitt House 28,
Southampton Buildings Chancery Lane
London WC2A 1AT (GB)

(54) Receiving buffer control system.

(57) A receiving buffer control system comprises a memory (4) having a buffer area (8) serving as a receiving buffer, data being applied to the memory via a bus, a write pointer (11) indicating a write address of the buffer area, and a read pointer (12) indicating a read address of the buffer area. An overrun/underrun detection circuit (17) detects a situation in which an overrun or an underrun will occur in the buffer area in response to the write address indicated by the write pointer and the read address indicated by the read pointer. A control part (9) disables the data from being written into and read out from the buffer area when the overrun/underrun detection circuit detects the situation.

FIG. 3



EP 0 507 571 A2

BACKGROUND OF THE INVENTION

(1) Field of the Invention

The present invention generally relates to a receiving buffer which receive data in a message communication, and more particularly to a system for controlling such a receiving buffer.

In a parallel computer system in which a plurality of processors are connected to each other via a network, each of the processors separately holds data. When a processor needs to use data held in another processor, data is transferred in a message communication. In the message communication, it is desired that data be transferred at a high as possible speed. In order to speed up the message communication, it is necessary to efficiently perform data buffering. The present invention is intended to realize a high-speed buffering system.

(2) Description of the Prior Art

Fig.1 shows a conventional parallel computer system, which comprises a plurality of processors 1 and 1' and a network 2. Each of the processors 1 and 1' comprises a CPU (Central Processing Unit) 3, a main memory 4, a DMA (Direct Memory Access) controller 5, a communication device 6, a common bus controller 6a, and a receiving buffer 7. These structural parts are connected to each other via a common bus BUS. The network 2 makes it possible for arbitrary processors, such as the processors 1 and 1', to communicate with each other. The communication device 6 executes an interface control process for the message communication. Data received from the network 2 is written into the receiving buffer 7 formed in the main memory 4. Data transmitted to the network 2 is read out from a sending buffer formed in the main memory 4. In order to simplify the explanation about the data receiving operation given below, only the receiving buffer 7 formed in the main memory 4 is illustrated in Fig.1. The DMA controller 5 executes, instead of the CPU 3, a buffer access control process for the data write and read operations. Parameters necessary for the buffer access control process are set by an instruction executed by the CPU 3.

The data receiving operation in the conventional message communication comprises the following first through fifth steps. In the first step of the data receiving operation, a sending processor (processor 1', for example) generates a message, which is transferred via the network 2 and received by a receiving processor (processor 1, for example). In the second step, the communication device 6 of the receiving processor 1 sends an interruption request to the CPU 3, and thereby informs the CPU 3 of the arrival of the message. In the third step, the CPU 3 acquires the receiving buffer 7 in the main memory 4. In the fourth step, the CPU 3 sets parameters, such as an address of the receiving buffer 7 and a received data length, in the DMA controller 5. In the fifth step, the data receiving operation is started while the CPU 3 and the DMA cooperate with each other.

In order to increase the speed of the above message communication, it is necessary to consider the time necessary to execute the second through fourth steps as well as the time necessary to interrupt the operation of the CPU 3 and execute a corresponding process when the receiving buffer 7 overruns. As the times directly relate to the operation of the CPU 3, they cannot be greatly reduced if the communication device 6 is designed to operate at a higher speed.

In order to reduce the above disadvantage, a receiving buffer of a ring buffer type as shown in Fig.2 has been proposed. A fixed area in the ring buffer can be continuously and circularly accessed by means of a read pointer (RDP) and a write pointer (WTP). In this manner, the tolerance of the storage capacity of the receiving buffer can be increased, and the setup process of the receiving buffer can be simplified so that the overhead thereof can be reduced. If the write operation or the read operation of the ring buffer excessively advances, the receiving buffer will overrun or underrun.

Japanese Laid-Open Patent Publication No. 59-112327 proposes a ring buffer control system in which the high-order address of a buffer area is expanded, and an overrun of the ring buffer is neglected so that the address wraps-around. According to this proposal, it is possible to increase the buffer capacity. However, the proposed ring buffer control system does not have means for detecting an overrun and underrun of the ring buffer. Further, it is very difficult to identify an area of the ring buffer in which received data which has not yet been processed.

Japanese Laid-Open Patent Publication No. 60-178570 discloses a ring buffer realized by the combination of the CPU and the DMA. No special improvement in the ring buffer has been proposed, and the CPU is interrupted each time data is received. The above is implemented by software, which is not designed to substantially prevent the ring buffer from overrunning. Further, the performance (efficiency of processing) of the proposed ring buffer is slightly less than 1/2 because the CPU having a shared bus structure manages data and is interrupted each time data is received. The interruption procedure is implemented by software.

As described above, the conventional ring buffer control systems do not have special means for preventing

run is executed by the write pointer 11 and the register 12'.

An address input buffer 18 holds an address written by the CPU 3 (Fig.3). The address in the address input buffer 18 is selectively written into the pointers 11 and 12 and the base registers 15 and 16 under the control of the CPU 3.

The receiving buffer of the ring buffer type is formed in the buffer area 8 in the main memory 4 and an access mechanism including the pointers 11 and 12 and the base registers 15 and 16 in the DMA controller 50.

The logical address of the starting position of the buffer area 8 is held in the base register 15, and the read address after address conversion is held in the base register 16. When the accessing source is a user program, the logical address is used as the base address. When the accessing source is a system program, the real address is used as the base address. The base address is combined with the address in either the write pointer 11 or the read pointer 12, which functions as a relative address.

The write pointer 11 is referred to when received data is written into the buffer area 8. The address in the write pointer 11 is incremented by 1 when the write operation ends. The address in the read pointer 12 is updated by the content of the register 12' when the content of the register 12' is read. The content of the register 12' is equal to an address obtained by increasing, by one, the address in the read pointer 12 used in the immediately previous access for data reading.

The output address generator 10 selects, on the basis of whether the access mode is a logical address mode or a real address mode or whether the access is for the data write or data read, an output address from among the addresses of the pointers 11 and 12 and the registers 12', 15 and 16. During this selection, the buffer size is taken into account so that the updating of the address executed by increasing the address in the pointer 11 or 12 conforms to a circulating address change within a predetermined area. The output address generator 10 sends back, to the CPU 2 instead of address, information (NULL) showing there is no data to be read out from the buffer area 8 in response to a data request from the register 12'.

The sequencer 9 sends and receives control signals to and from the communication device 6 (Fig.1) and the common bus controller 6a. The sequencer 9 acquires the right to use the common bus BUS, and executes the access control in accordance with a data transfer request and a transfer completion notification signal. When the buffer area 8 overruns or underruns or when an error has occurred, the sequencer 9 stops the access control.

Initially, the address of the starting position of the buffer area 8 is written into the write pointer 11, and the address of the end position of the buffer area 8 is written into the read pointer 12. When the communication device 6 (Fig.1) requests to transfer data, the DMA controller 50 sequentially writes the received data in the buffer area 8 starting from the address specified by the write pointer 11. With this operation, it is possible for the processor to continuously execute an operation without being affected by a message transmission non-periodically generated by another processor. When a message is needed, the processor can immediately obtain the starting address of the area in which the needed data is stored or can immediately acknowledge that there is no data which is to be read. A decision on whether the buffer overruns or underruns is made by means of hardware. When the buffer overruns, the processor is notified of the overrun by an interruption procedure, or the data transfer is stopped until a receiving area becomes available in the buffer area 8 by executing the data read operation. It is possible for the user to select one of the above procedures.

It is possible to obtain necessary information about the status of the buffer by reading the contents of the pointers 11 and 12 and the register 12' which manage the buffer. The starting position of data which has not yet been read can be identified from the read address specified by the pointer 12. It is possible to determine up to which position of the buffer data to be read exists by reading the write address specified by the write pointer 11. By reading the address in the register 12', the area accessed immediately before being specified by the read pointer 12 can be released, and the address specified by the read pointer 12 is replaced with the address (RDP+1) in the register 12'.

Fig.4 shows the DMA controller 50 shown in Fig.3 in more detail. The output address generator 10 comprises multiplexers (MPX) 19 - 21. The overrun/underrun detection circuit 17 comprises comparators 22 and 23. During continuous accessing, the sequencer 9 generates a read pointer (RDP) updating signal incrdp, and a write pointer (WTP) updating signal dmai. Further, the sequencer 9 stops the access control in response to an overrun detection signal wmatch. The write pointer 11 and the read pointer 12 are respectively incremented by 1 in response to the WTP updating signal dmai and the RDP updating signal incrdp.

The comparator 22 in the overrun/underrun detection circuit 17 compares the current address specified by the write pointer 11 with the current address specified by the read pointer 12. When these addresses match each other, the comparator 22 generates the overrun detection signal wmatch, and thereby informs the sequencer 9 that an overrun of the receiving buffer will occur during the next write operation.

The comparator 23 compares the address obtained by incrementing the address specified by the read pointer 12 by one with the current address specified by the write pointer 11. When these addresses match each

the ring buffer from overrunning and underrunning.

SUMMARY OF THE INVENTION

It is a general object of the present invention to provide a ring buffer control system in which the above disadvantages are eliminated.

A more specific object of the present invention is to provide a ring buffer control system which has a hardware means for effectively preventing a ring buffer from overrunning and underrunning.

The above objects of the present invention are achieved by a receiving buffer control system comprising: a memory having a buffer area serving as a receiving buffer, data being applied to the memory via a bus; a write pointer, coupled to the memory, for indicating a write address of the buffer area; a read pointer, coupled to the memory, for indicating a read address of the buffer area; an overrun/underrun detection circuit, coupled to the write and read pointers, for detecting a situation in which an overrun or an underrun will occur in the buffer area in response to the write address indicated by the write pointer and the read address indicated by the read pointer; and a control part, coupled to the memory and the overrun/underrun detection circuit, for preventing the data from being written into and read out from the buffer area when the overrun/underrun detection circuit detects the situation.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects, features and advantages of the present invention will become more apparent from the following detailed description when read in conjunction with the accompanying drawings, in which:

Fig.1 is a block diagram of a conventional parallel processor system;
Fig.2 is a block diagram illustrating a conventional ring buffer;
Fig.3 is a block diagram illustrating an outline of an embodiment of the present invention;
Fig.4 is a block diagram of a DMA controller shown in Fig.3;
Fig.5 is a block diagram of an output address generator shown in Fig.3;
Fig.6 is a block diagram of a data size multiplexer in the output address generator shown in Fig.5;
Fig.7 is a circuit diagram of an overrun/underrun detection circuit shown in Fig.3;
Fig.8 is a diagram showing the state transition of a sequencer shown in Fig.3;
Fig.9 is a diagram showing an initialized state of a receiving buffer;
Fig.10 is a diagram showing a process of continuously accessing split data;
Fig.11A is a block diagram of a conventional cache data invalidating circuit;
Fig.11B is a block diagram of a cache data invalidating circuit used in the embodiment;
Fig.12 is a block diagram showing a process of invalidating cache data according to the embodiment of the present invention;
Fig.13A is a diagram showing how addresses are processed in order to continuously read split data;
Fig.13B is a block diagram of a structure for executing the address processing shown in Fig.13A; and
Fig.14 is a diagram of a structure used when a plurality of receiving buffers are employed.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig.3 shows the outline of an embodiment of the present invention. A DMA controller 50 is substituted for the DMA controller 5 shown in Fig.1. The DMA controller 50 is configured as follows. A sequencer 9 controls the operations of the structural parts of the DMA controller 50 in accordance with predetermined sequences. An output address generator 10 generates a memory access address. A write pointer (WTP) 11 specifies the next write address. A read pointer (RDP) 12 specifies the next read address. A register (RDPP) 12' holds an advanced read address (RDP+1) which precedes, by one, to the read address specified by the read pointer 12. An adder 13 increments, by one, the read address specified by the read pointer 12. An adder 14 increments, by one, the write address specified by the write pointer 11. A logical address base register (LAR) 15 holds the starting address of a buffer area 8 formed in the main memory 4 in the form of a logical (virtual) address. A real address base register (PAR) 16 holds the starting address of the buffer area 8 in the form of a real (physical) address.

An overrun/underrun detection circuit 17 compares the write address specified by the write pointer 11 with the read address specified by the read pointer 12 as well as the address registered in the register 12'. When the write address matches either the address specified by the read pointer 12 or the address in the register 12', the overrun/underrun detection circuit 17 concludes that the buffer area 8 will overrun or underrun. The detection of the overrun is executed by the write pointer 11 and the read pointer 12. The detection of the under-

other, the comparator 23 generates an underrun detection signal rmatch.

The multiplexer 19 selects either the register 15 or the register 16 in accordance with a control signal (not shown for the sake of simplicity), which shows whether the accessing source is a user program or a system program. In this manner, a predetermined number of high-order bits of the output address are determined. The multiplexer 20 selects one of the read pointer 12, the register 12', the write pointer 11 and 0 in accordance with control signals, such as the RDP updating signal incrdp, the WTP updating signal dmai and the underrun detection signal rmatch. In this manner, a predetermined number of low-order bits of the output address are determined.

The multiplexer 21 has a decoder function of defining the receiving buffer size. In accordance with a specified size signal SIZE, the multiplexer 21 selects a predetermined number of middle-order bits out of the address bits from the multiplexer 19 or a predetermined number of middle-order bits out of the address bits from the multiplexer 20. The buffer size defined by the middle-order bits, the output address endlessly changes in the buffer size unit even when the addresses of the pointers 11 and 12 are incremented by 1 in a direction.

Fig.5 shows the output address generator 10 in more detail. It will now be assumed that the bit width of data from each of the registers (LAR) 15 and (PAR) 16 is equal to 20 bits, and the bit width of data from each of the read pointer (RDP) 12, the register (RDPP) 12' and the write pointer (WTP) 11 is equal to 15 bits. Symbols Srdp, Srdpp and Swtp are select control signals which are generated by the sequencer 9 and applied to the RDP, RDPP and WTP, respectively. A symbol "dma" denotes the DMA access, "system" denotes an access from the system program, and "lar" denotes an access from the user program using the logical address.

When both the signals Srdpp and rmatch are 'H (high)', the multiplexer 19 generates 0. When either the signal dma or system is 'H' in cases other than the case where both the signal Srdpp and rmatch are 'H', the multiplexer 19 generates the real address PAR. When the signal lar is 'H' in cases other than the case where both the signal Srdpp and rmatch are 'H', the multiplexer 19 generates the logical address LAR.

The multiplexer 20 selects the read address RDP specified by the read pointer 12 when the signal Srdp is 'H', and selects the address RDPP in the register 12' when both the signals Srdpp and rmatch are 'H'. Further, the multiplexer 20 selects the write address specified by the write pointer WTP when either the signal Swtp or dma is 'H', and selects 0 when both the signals Srdpp and rmatch are 'H'.

The multiplexer 21 defines the buffer size. As shown in Fig.6, the multiplexer 21 comprises a decoder (DEC) 25 and multiplexers (MPX) 26-0 - 26-6. The decoder 25 converts three-bit data showing the buffer size into seven signals SIZE0, SIZE1-0, SIZE2-0, ..., SIZE6-0, which are applied to the multiplexers 26-1 - 26-6, respectively. The signal SIZE0 output by the decoder 25 is 'H' only when the buffer size data SIZE indicates 0. The signal SIZE1-0 is 'H' when the buffer size data SIZE indicates either 1 or 0. The signal SIZE2-0 is 'H' when the buffer size data SIZE is one of 2, 1 and 0. Generally, a signal SIZEi-0 (where i = 0, 1, 2, ..., 6) is 'H' when the buffer size data SIZE indicates a value smaller than or equal to the value i. The multiplexers 26-0 - 26-6 respectively receive seven low-order bits AR(6) - AR(0) from the multiplexer 19 and seven high-order bits PTR(6) - PTR(0). Each multiplexer 26-i selects the bit on the AR side when the signal SIZEi-0 is 'H'.

Returning now to Fig.5, the high-order, middle-order and low-order address bits respectively output from the multiplexers 19, 20 and 21 are combined with each other in the buffer 24, and output as an address AD.

Figs.7A, 7B and 7C show the overrun/underrun detection circuit 17 in detail. Fig.7A shows the configuration of each of the comparators 22 and 23. Each comparator comprises 15 ENOR (Exclusive-NOR) circuits, which respectively compare the 15 bits of the write address WTP with the 15 bits of the read address RDP. Output signals of the 15 ENOR circuits are denoted by eq(0) - eq(14).

Fig.7B shows a decoder which decodes the buffer size signal SIZE. The decoder shown in Fig.7B converts the three bits of the signal SIZE into seven signals SIZE0 - SIZE6

Fig.7C shows a detection signal generator which generates the overrun detection signal wmatch or the underrun detection signal rmatch. In Fig.7C, a signal "match" denotes either the signal wmatch or rmatch. An AND circuit AND1 detects a status where all the low-order bits eq(7) - eq(0) are 'H'. A plurality of AND circuits AND2 receive a predetermined number of high-order bits out of the high-order bits eq(14) - eq(8) and a corresponding one of the signals SIZEi. This is intended to change the comparison range on the basis of the specified buffer size SIZEI. Output signals of the AND circuits AND2 are applied to an OR circuit, an output signal of which is applied to an AND circuit AND3. An output signal of the AND circuit AND1 is applied to the AND circuit AND3. An output signal of the AND circuit AND3 is the detection signal wmatch or rmatch.

Fig.8 is a diagram showing a state transition controlled by the sequencer 9 shown in Fig.3. In Fig.8, "IDLE" denotes an operation halt state, and "BWAIT", BADDR" and "BWAIT" denote operating states. When an error has occurred in the states "BWAIT", BADDR" and "BWAIT", the state returns to the state IDLE. In the state BWAIT, the communication device 6 (Fig.1) becomes ready to transfer data, generates the data transfer request when the receiving buffer has a sufficient idle area, and waits for acquisition of the bus.

When the bus has been acquired in the state BWAIT, so that data is allowed to be transferred, the sequ-

encer 9 switches to the state BADDR. In this state, an address is output to the address bus. After one clock from this address output operation, the sequencer 9 switches to the state BDATA, in which the sequencer 9 waits for completion of the data transfer. If a subsequent data transfer request has been generated and the right to use the bus has been acquired after the completion of the data transfer is acknowledged, the sequencer 9 returns to the state BADDR, in which data is continuously transferred. If there is no data transfer request, the sequencer 9 returns to the state BWAIT, and waits for another data transfer request.

A description will now be given of the operation of the embodiment of the present invention.

As shown in Fig.9, when the CPU 3 (Fig.1) reads the address in the register (RDPP) 12' in the state where the write pointer (WTP) 11 indicates the top of the receiving buffer and the read pointer (RDI) 12 indicates the bottom thereof, 0 (NULL) is sent back to the CPU 3 since there is no data in the receiving buffer. In Fig.5, the following is obtained:

$$Srdpp = 1,$$
$$rmatch = CMP(RDP + 1, WTP) = 1.$$

Hence the output address is zero (AD = 0). In this manner, it is easy for the CPU 3 to determine whether or not there is data in the receiving buffer.

An example of use of the receiving buffer will now be described. A description will now be given of a case where an area of 8Kbyts from real address OX' 5A000 to real address OX' 5BFFF is used. In this state, it will now be assumed that the memory space is mapped from address OX' 82000. In order to distinguish the system mode and the user mode from each other, "sys" or "usr" is placed in advance of RDP and WTP.

The LAR, PAR, RDP and WTP are set as follows:

$$LAR \leftarrow OX' 82000$$
$$PAR \leftarrow OX' 5A000$$
$$RDP \leftarrow OX' 01FF0$$
$$WTP \leftarrow OX' 00000.$$

The RDP and WTP are initialized when they are reset.

A command register BCMD, which is a control register always provided for the DMA controller, is set. Similarly, a status register BRST is provided.

$$\text{For SIZE} = 110, WC = 0 \text{ and } ST = 1,$$
$$BCMD \leftarrow OX' 86000000.$$

When the read address (RDP) of the read pointer 12 and the write address (WTP) of the write pointer 11 are read, the following is obtained:

$$\text{sys RDP} = OX' 5BFF0$$
$$\text{usr RDP} = OX' 83FF0$$
$$\text{sys WTP} = OX' 5A000$$
$$\text{usr WTP} = OX' 82000.$$

However, since the receiving buffer does not have data, when the register (RDPP) 12' is read, the following is sent back:

$$\text{sys RDPP} = OX' 00000$$
$$\text{usr RDPP} = OX' 00000.$$

When data equal to three blocks (16bytes per block) is received, the write pointer WTP is sequentially updated in the following manner, and the received data is written into the receiving buffer:

sys OX' 5A000
sys OX' 5A010
sys OX' 5A020.

When all the data has been completely received, the write address WTP of the write pointer 11 is OX' 5A030.

At this time, when the content RDPP of the register 12' is read, the following information is sequentially obtained:

usr RDPP = OX' 82000 (which shows that data of the first block is placed at the user address OX' 82000 (system address OX' 5A000));
usr RDPP = OX' 82010 (which is the user address of data of the second block);
usr RDPP = OX' 82020 (which is the user address of data of the third block);
usr RDPP = OX' 00000 (which shows there is no data); and
usr RDPP = OX' 00000 (which shows there is no data). In this manner, the user program (CPU 3) can recognize that there is no data subsequent to the three blocks.

When data equal to 8Kbytes has been received and hence the receiving buffer has been filled with the data, the receiving buffer overruns and the overrun detection signal wmatch becomes equal to 1 (wmatch = 1). Thus, the sequencer 9 (Fig.3) temporarily halts the data transfer in order to prevent data from being lost.

The read address RDP of the read pointer 12 is updated by the content RDPP of the register 12'. When

an idle area in the receiving buffer becomes available, the data receiving operation is restarted.

If data which is recognized to be unnecessary beforehand is contained in the received data, the read pointer 12 is rewritten so that the read operation jumps some data. For example, when the read address RDP and the write address WTP are as follows:

sys RDP = OX' 5A010

sys WTP = OX' 5A810

the following is written into the read pointer RDP in order to jump data equal to 16 blocks:

sys RDP ← OX' 5A110.

The writing operation on the register is performed via the bus which is also used for the DMA transfer operation. Hence, the DMA controller 50 does not execute the next operation until the above jumping operation is completed.

As shown in Fig.10, when the received data requested to be read is split into two areas as follows:

OX' 5BF00 - OX' 5BFFF

OX' 5A000 - OX' 5A0FF

it is possible to continuously access these areas by doubling the buffer area, as shown by a broken line in Fig.10.

There is a possibility that the data written into the main memory will not coincide with data written into a cache memory. By invalidating the data in the cache memory by using the address in the register (RDPP) 12', it becomes possible to ensure the matching between data in the main memory and data in the cache memory.

A process of invalidating data in the cache memory will now be described with reference to Figs.11A, 11B and 12. Fig.11A shows a conventional cache data invalidating circuit, and Fig.11B shows a cache data invalidating circuit according to the embodiment of the present invention. The circuit shown in Fig.11A comprises a tag memory (TAG) 27, an attribute memory (ATTR) 28, a cache data memory 29, a comparator 30, and an AND gate 31. The address ADD is split into a high-order address and a low-order address. The tag memory 27, the attribute memory 28 and the cache data memory 29 are simultaneously accessed by the low-order address. When the content of the tag memory 27 is the same as the high-order address and the content of the attribute memory 28 indicates "valid", a hit signal is output by the AND gate and cache data is used. If the content of the tag memory 27 does not coincide with the high-order address or the content of the attribute memory 28 indicates "invalid", data is read out from the main memory. The data is held in the cache memory until data having the same tag as that of the data in the cache memory.

When data is written into the main memory with respect to the address of data held in the cache memory by an I/O process of data receiving and writing, data in the main memory becomes different from data in the cache memory.

As shown in Fig.12(a), data C is stored in an area of a main memory 35 specified by address B. When the data C is read out from the main memory 35, the data C is held in a cache memory 34. Thereafter, when data D is written into the area of the main memory 35 specified by address B, as shown in Fig.12(b), the data in the cache memory 34 does not coincide with the data in the main memory 35. When the cache memory 34 is accessed in order to read the data D therefrom, the data C is read out therefrom since the data C in the cache memory 34 specified by address B is still valid. That is, data D cannot be read out from the cache memory 34.

Hence, it is necessary to invalidate data in the cache memory 34 specified by the updated address of the main memory 35. According to the embodiment of the present invention, when the data read operation is executed after received data is written in the receiving buffer in the main memory, the address RDPP in the register 12' is read and then output to the bus. The data in the cache memory 34 is invalidated by using the above address (see Fig.12(b)).

Fig.11B shows the cache data invalidating circuit which realizes the above procedure. The circuit shown in Fig.11B comprises multiplexers 32 and 33. The multiplexer 32 receives a high-order part of the address RDPP which is read out from the register 12' and output to the common bus by the DMA controller 50. The multiplexer 33 receives a low-order part of the above address RDPP. Further, the multiplexers 32 and 33 receive the high-order and low-order parts of the address ADD, respectively. When the address RDPP is read out from the register 12', the high-order and low-order parts of the address RDPP are respectively selected by the multiplexers 32 and 33 instead of the high-order and low-order parts of the address ADD. Then, the tag memory 27 and the attribute memory 29 are accessed by the selected address. When the hit signal is output, the invalid signal is written into the attribute memory 28, so that data in the cache memory 34 can be invalidated.

A description will now be given, with reference to Figs.13A and 13B, of a circuit capable of continuously accessing data split in upper and lower part of the receiving buffer. As shown in Fig.13A, a mapping is carried out so that addresses spaced apart from each other in a real memory 37 become consecutive addresses. An address area (address image) twice the size of the real buffer area is provided in order to avoid the separation of addresses specifying the received data.

For example, addresses 5C000 - 5E000 of an address image 36 are placed in addresses 5A000 - 5C000

of the real memory 37. By accessing the addresses 5BF00 - 5C100, it becomes possible to continuously read out data split in the real memory 37. Fig.13B shows an address conversion circuit which implements the above address conversion. An address bit AD[14:13] is converted into '01' between 5C000 - 5E000. A decoder (DEC) 38 detects a section in which the address conversion should be performed, and a multiplexer (MPX) 39 converts the bit value when the section is detected.

A description will now be given, with reference to Fig.14, of a structure in which a plurality of receiving buffers are provided, these buffers being selectively used on the basis of the type of received data and/or the type of the accessing source. As shown in Fig.14(a), a plurality of sets of the registers RDP, WTP, LAR, PAR, BCMD and BRST are provided and selectively used on the basis of the message type and/or a task ID.

The message type can be extracted from a header of the received data, and the task ID can be obtained in a register which holds the task ID. Fig.14(b) is a detailed view of the structure having a plurality of receiving buffers. As shown in this figure, a plurality of register groups 40 - 42 relating to the respective types of, for example, messages, are chained and further connected to a decoder (DEC) 44. A register 43, which is connected to the decoder 44, holds data showing a desired type of message. The above data is written into the register 43 by, for example, the CPU 3 (Fig.1). The decoder 44 decodes the data in the register 43, and selects one of the register groups 40 - 42. Each of the register groups 40 - 42 is assigned a plurality of addresses in order to selectively read out the data in the registers RDP, WTP, LAR, PAR, BCMD and BRST. With the above structure, it becomes possible to efficiently access only a specific message by a single interface.

The present invention has the following advantages. First, it becomes possible to omit a series of processes executed by the CPU after data arrives at the processor. Hence, it becomes possible to reduce the setup time after data is received and reduce the load of the CPU. Next, the pointers associated with the receiving buffer is automatically updated by hardware. Hence, it becomes possible to eliminate the problems arising from use of software. In addition since the data read operation can be jumped, data can be efficiently read at a high speed. Also, information showing that there is no data in the receiving buffer can be obtained by referring to the information in the registers for managing the receiving buffer. Hence, data can be efficiently read out from the receiving buffer. The page address is managed in the form of logical address and real address. Hence, it becomes possible to rapidly generate the memory address even when the access mode is switched. By selectively using a plurality of receiving buffers on the basis of the message type or the task ID, it becomes possible to efficiently access specific data. It becomes possible to automatically invalidate data about the address of the cache memory by using the address in the register for managing the receiving buffer. Hence, the content of the cache memory always matches data in the main memory. By providing an address area twice that of the receiving buffer, it becomes possible to continuously access split data and hence efficiently execute the access operation.

The present invention is not limited to the specifically disclosed embodiments, and variations and modifications may be made without departing from the scope of the present invention.

Claims

1. A receiving buffer control system comprising:

   a memory (4) having a buffer area serving as a receiving buffer (8), data being applied to said memory via a bus;

   write pointer means (11), coupled to said memory, for indicating a write address of the buffer area; and

   read pointer means (12), coupled to said memory, for indicating a read address of the buffer area,

   characterized in that said receiving buffer control system comprises overrun/underrun detection means (17), coupled to said write pointer means and said read pointer means, for detecting a situation in which an overrun or an underrun will occur in the buffer area in response to the write address indicated by said write pointer means and said read address indicated by said read pointer means; and

   control means (9), coupled to said memory and said overrun/underrun detection means, for disabling the data from being written into and read out from the buffer area when said overrun/underrun detection means detects said situation.

2. A receiving buffer control system as claimed in claim 1, characterized by further comprising updating means (12') for holding an advanced read address which precedes, by a predetermined address value, to the read address indicated by said read pointer means and for updating the read address indicated by said read pointer means by said advanced read address.

3. A receiving buffer control system as claimed in claim 1, characterized by further comprising means (3) for

writing an external read address into said read pointer means,said external read address indicating split data in the buffer area so that said split data are continuously read out from the buffer area.

4. A receiving buffer control system as claimed in claim 2, characterized by further comprising means (3) for determining, on the basis of the advanced read address, whether or not there is data in the buffer area.

5. A receiving buffer control system as claimed in claim 1, characterized in that:
said receiving buffer control system further comprises register means (15, 16) for holding a first base address of a logical address type and a second base address of a real address type;
each of said first and second base addresses is used for updating the write address indicated by the write pointer means and the read address indicated by the read pointer means; and
either said first base address or said second base address is used on the basis of the type of data input to the buffer area.

6. A receiving buffer control system as claimed in claim 1, characterized in that:
said memory comprises a plurality of buffer areas; and
said receiving buffer control system comprises means for detecting the type of data and for selecting one of the plurality of buffer areas on the basis of the type of data.

7. A receiving buffer control system as claimed in claim 6, characterized in that:
said write pointer means comprises a plurality of write pointers (WTP0, WTP1, WTP2) respectively provided for said buffer areas;
said read pointer means comprises a plurality of read pointers (RDP0, RDP1, RDP2) respectively provided for said buffer areas;
said overrun/underrun detection means detects, for each of said buffer areas, said situation; and
said control means disables, for each of said buffer areas, the data from being written into and read out from the buffer area when said overrun/underrun detection means detects said situation.

8. A receiving buffer control system as claimed in claim 7, characterized by further comprising means (43, 44) for selecting one of the write pointers and one of the read pointers on the basis of the type of data.

9. A receiving buffer control system as claimed in claim 1, characterized by further comprising:
a cache memory (34) into which data identical to the data written into said buffer area is written; and
means (27-33), operatively coupled to said updating means, for invalidating the data in the cache memory when said advanced read address is output.

10. A receiving buffer control system as claimed in claim 2, characterized in that said updating means comprises a register (12') storing the advanced read address.

FIG. 1 PRIOR ART

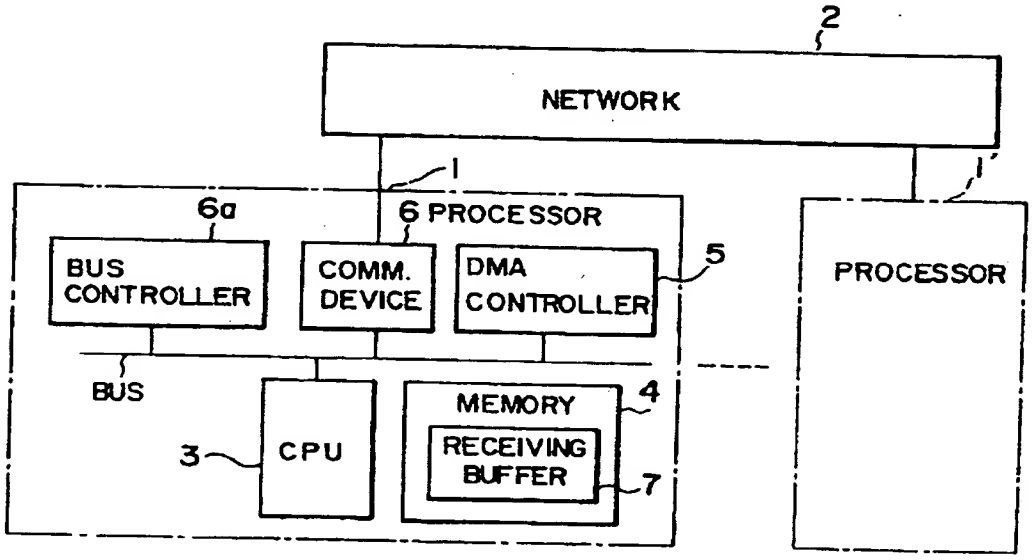


FIG. 2 PRIOR ART

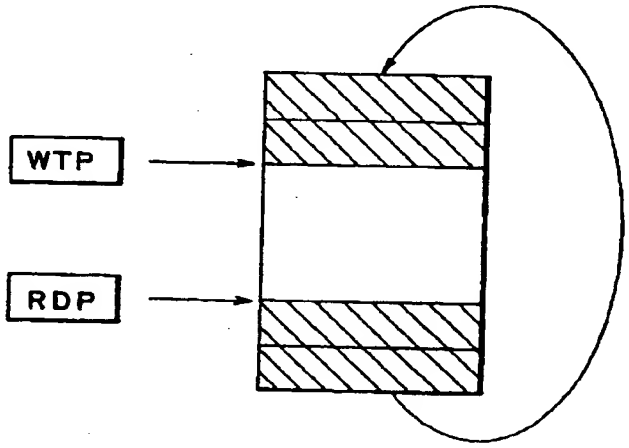


FIG. 3

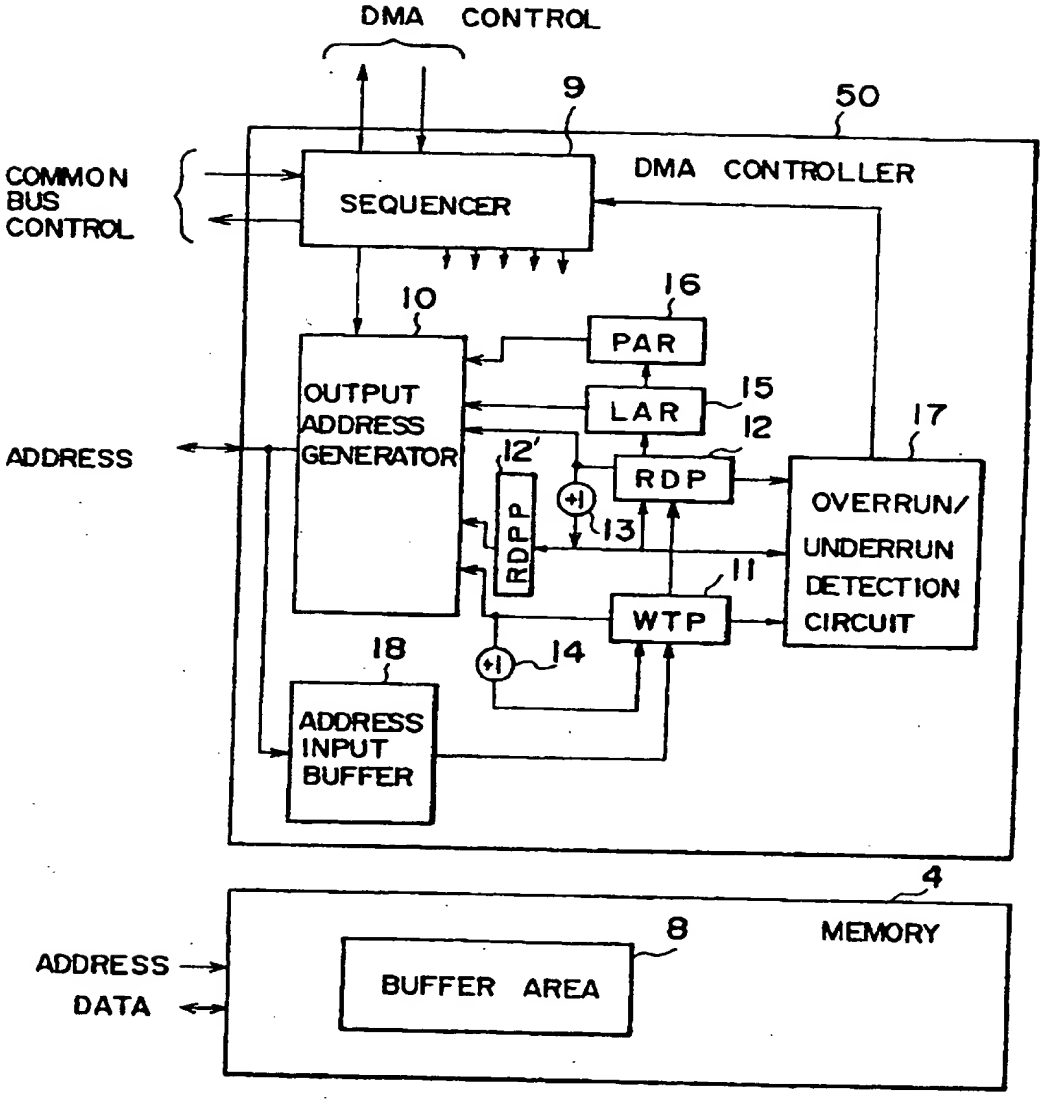

DMA CONTROL
9
50
COMMON BUS CONTROL
SEQUENCER
DMA CONTROLLER
10
16
PAR
OUTPUT ADDRESS GENERATOR
LAR
15
12
17
ADDRESS
12'
RDP
RDPP
+1
13
OVERRUN/ UNDERRUN DETECTION CIRCUIT
11
WTP
18
+1
14
ADDRESS INPUT BUFFER
4
MEMORY
8
ADDRESS
DATA
BUFFER AREA

FIG. 4

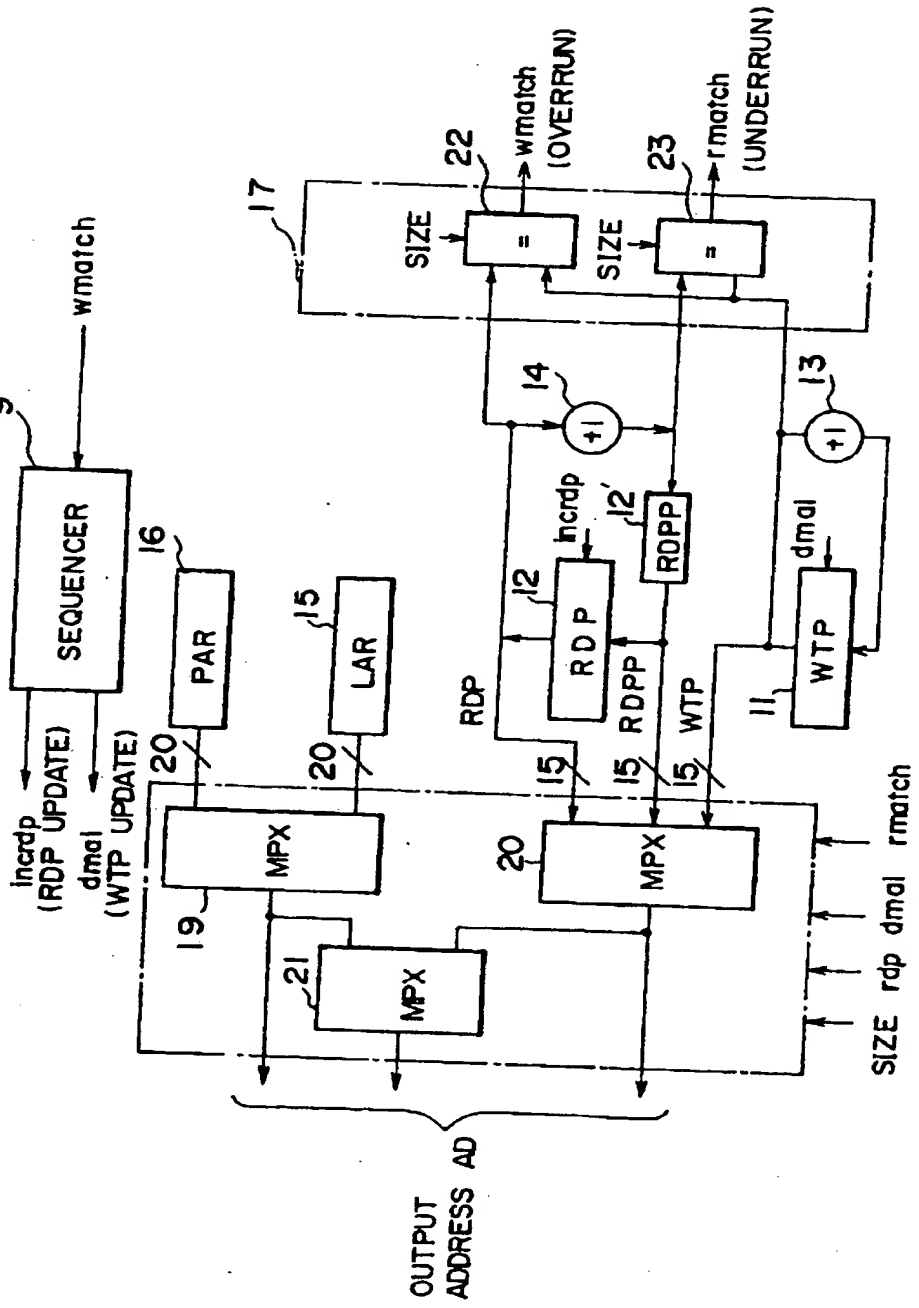

9
SEQUENCER
wmatch
16
PAR
15
LAR
20
20
MPX
19
21
MPX
20
MPX
incrdp
(RDP UPDATE)
dmal
(WTP UPDATE)
OUTPUT
ADDRESS AD
17
SIZE
=
22
wmatch
(OVERRUN)
SIZE
=
23
rmatch
(UNDERRUN)
14
+1
13
+1
incrdp
12
RDP
12'
RDPP
dmal
RDP
RDP
RDPP
WTP
WTP
11
15
15
15
SIZE rdp dmal rmatch

FIG. 5

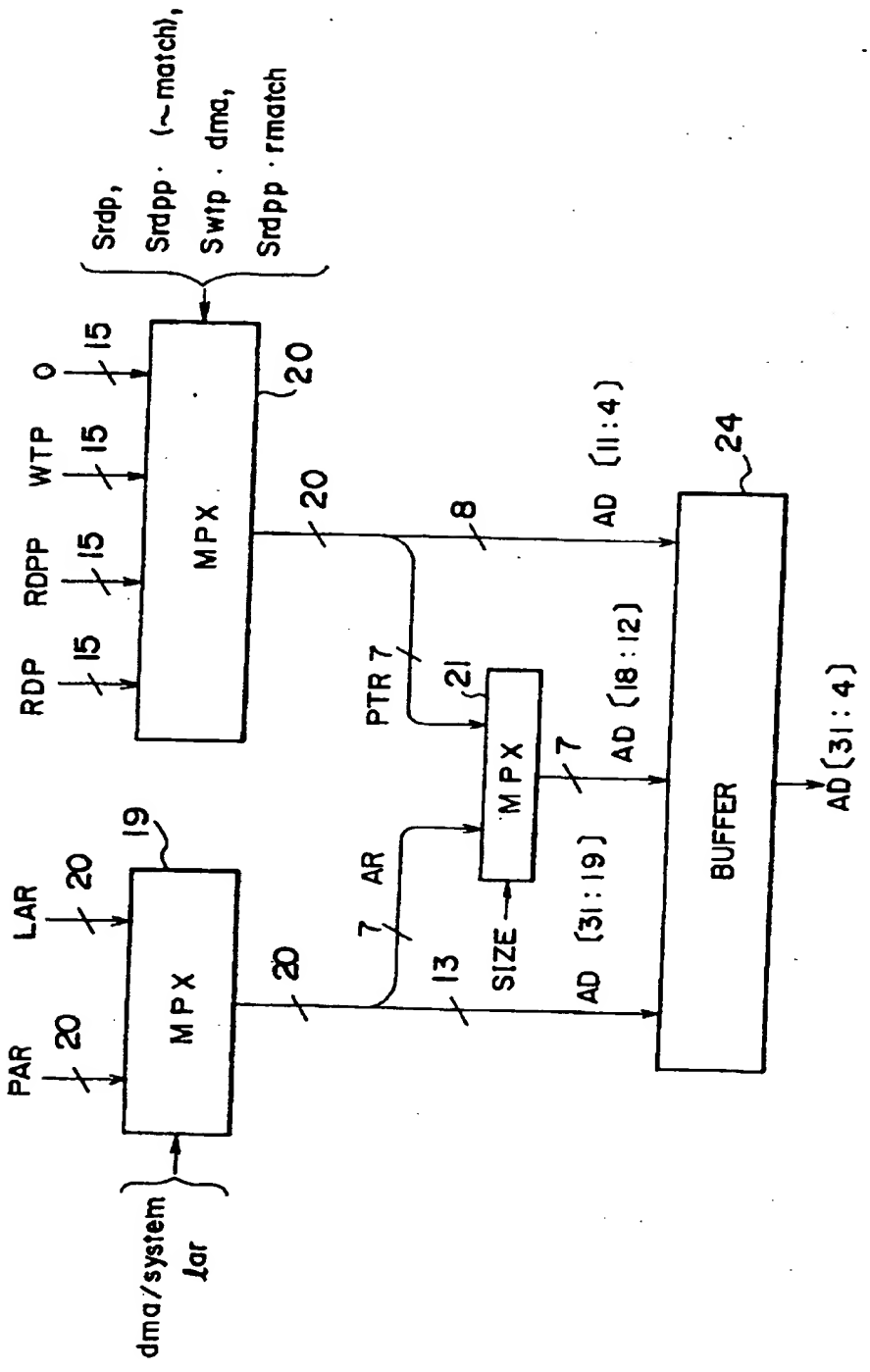

Srdp,
Srdpp · (~match),
Swtp · dma,
Srdpp · rmatch
RDP
RDPP
WTP
0
15
15
15
15
MPX
20
20
8
AD (11:4)
24
PTR 7
21
MPX
7
AD (18:12)
BUFFER
AD(31:4)
LAR
20
19
PAR
20
MPX
20
7
AR
13
SIZE
AD (31:19)
dma/system
lar

FIG. 6

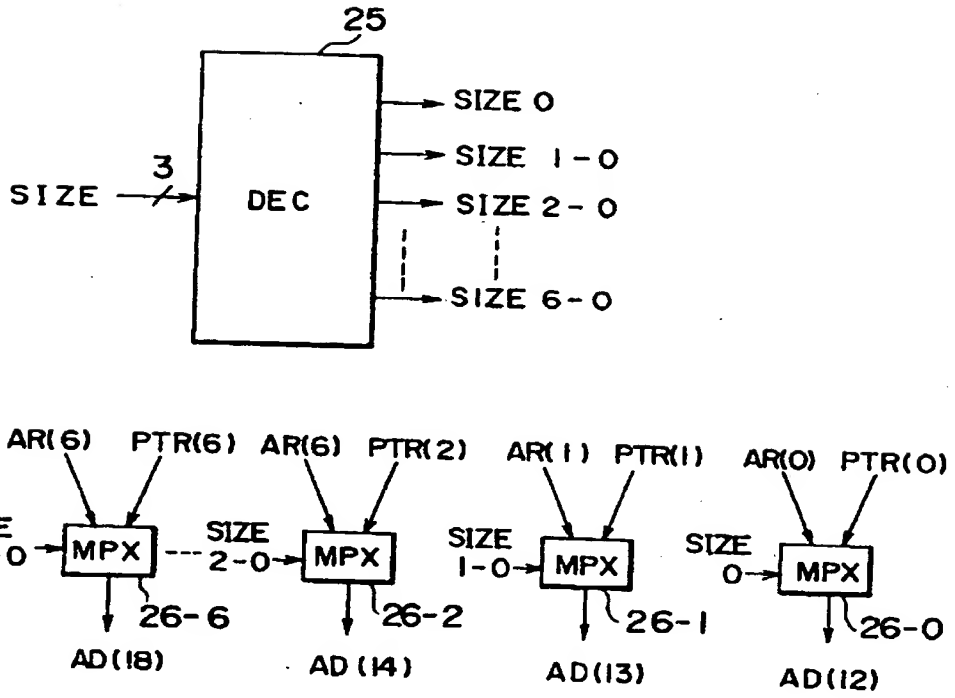

25
SIZE
3
DEC
SIZE 0
SIZE 1-0
SIZE 2-0
SIZE 6-0
AR(6)
PTR(6)
AR(6)
PTR(2)
AR(1)
PTR(1)
AR(0)
PTR(0)
SIZE 6-0
MPX
SIZE 2-0
MPX
SIZE 1-0
MPX
SIZE 0
MPX
26-6
26-2
26-1
26-0
AD(18)
AD(14)
AD(13)
AD(12)

FIG. 7A

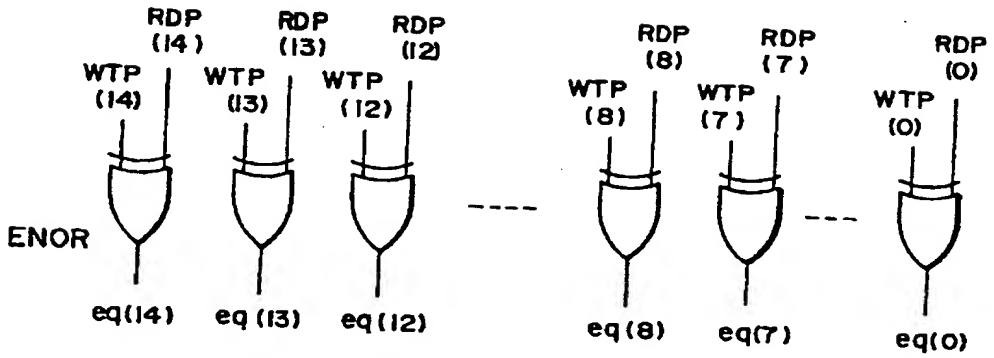


FIG. 7B

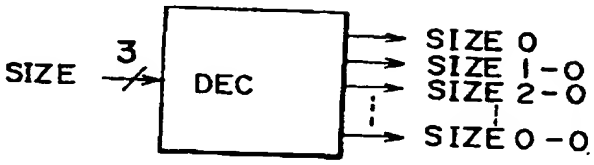


FIG. 7C

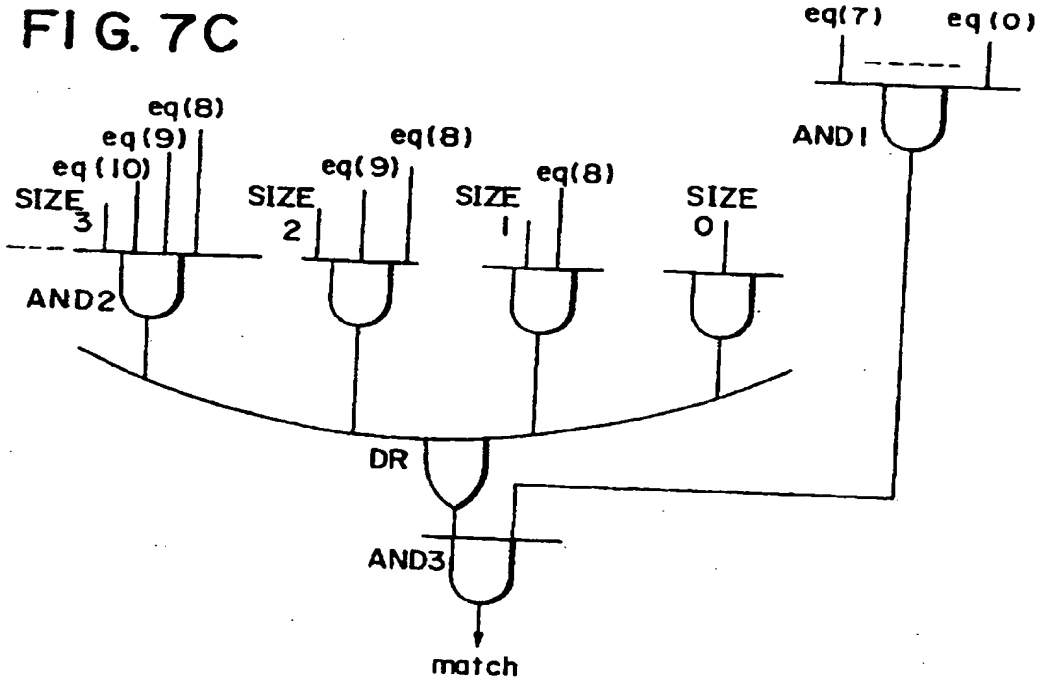


FIG. 8

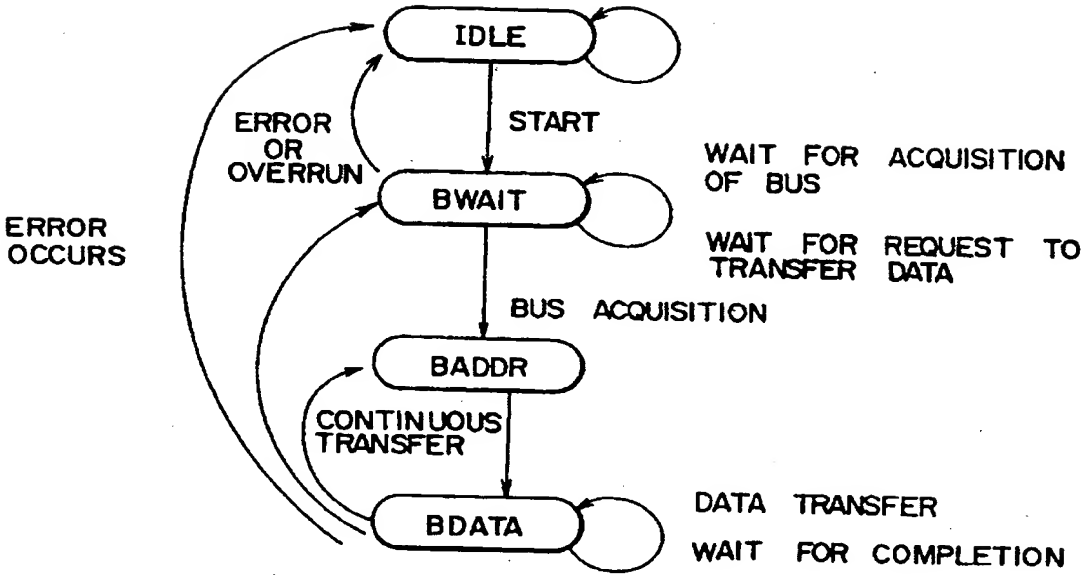

IDLE
START
ERROR OR OVERRUN
BWAIT
WAIT FOR ACQUISITION OF BUS
ERROR OCCURS
WAIT FOR REQUEST TO TRANSFER DATA
BUS ACQUISITION
BADDR
CONTINUOUS TRANSFER
BDATA
DATA TRANSFER
WAIT FOR COMPLETION

FIG. 9

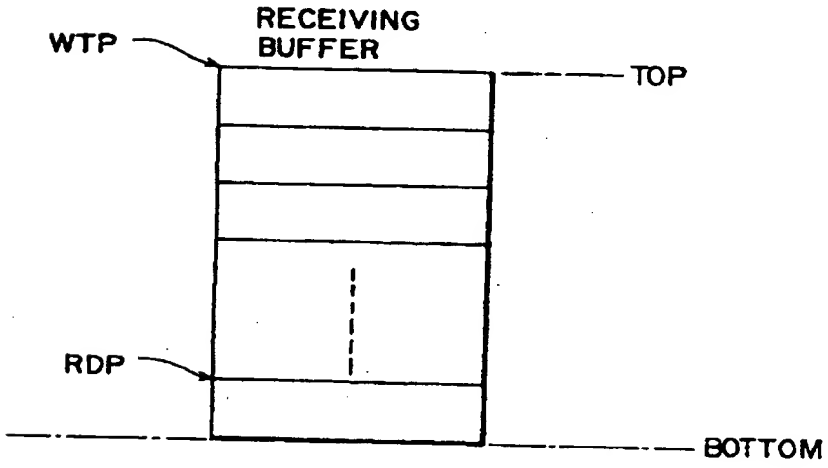

WTP
RECEIVING BUFFER
TOP
RDP
BOTTOM

FIG. 10

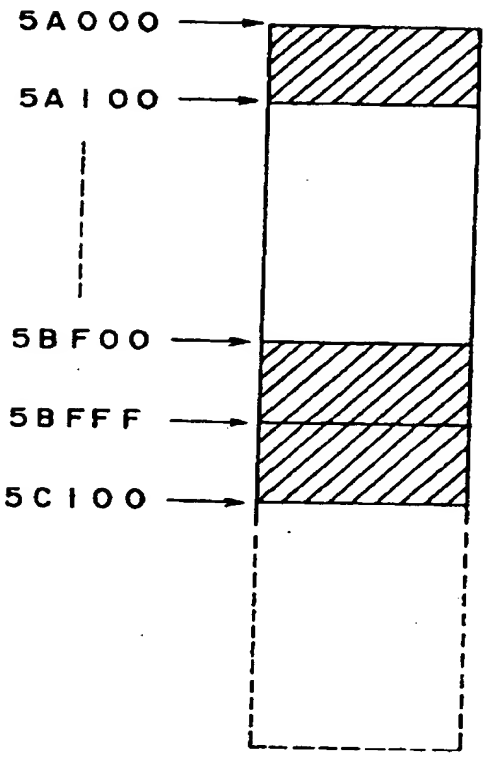


FIG.11A PRIOR ART

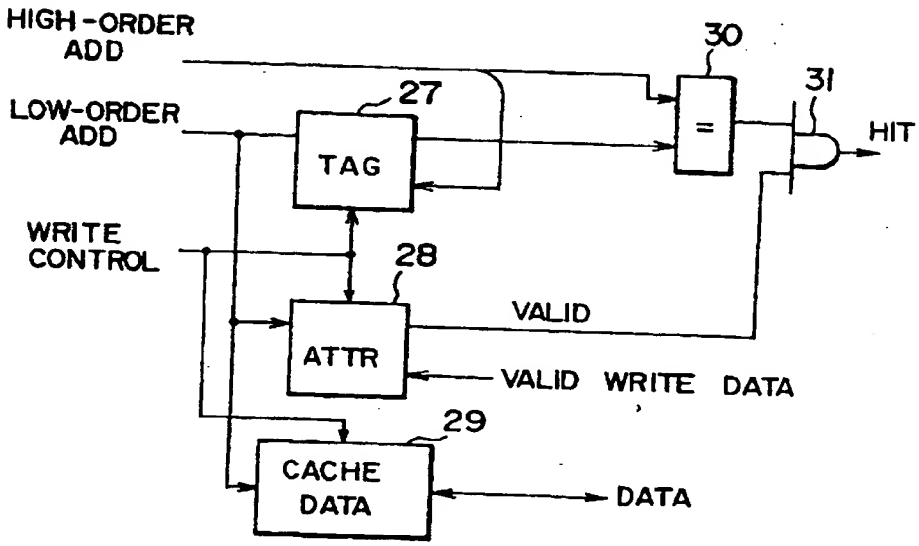


FIG. 11B

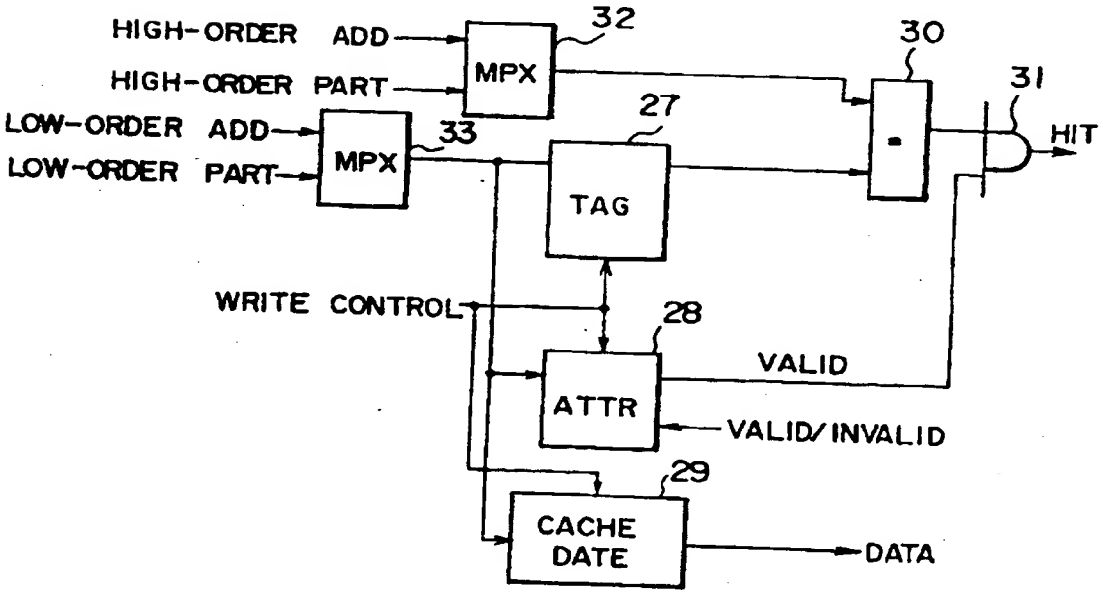


FIG. 12

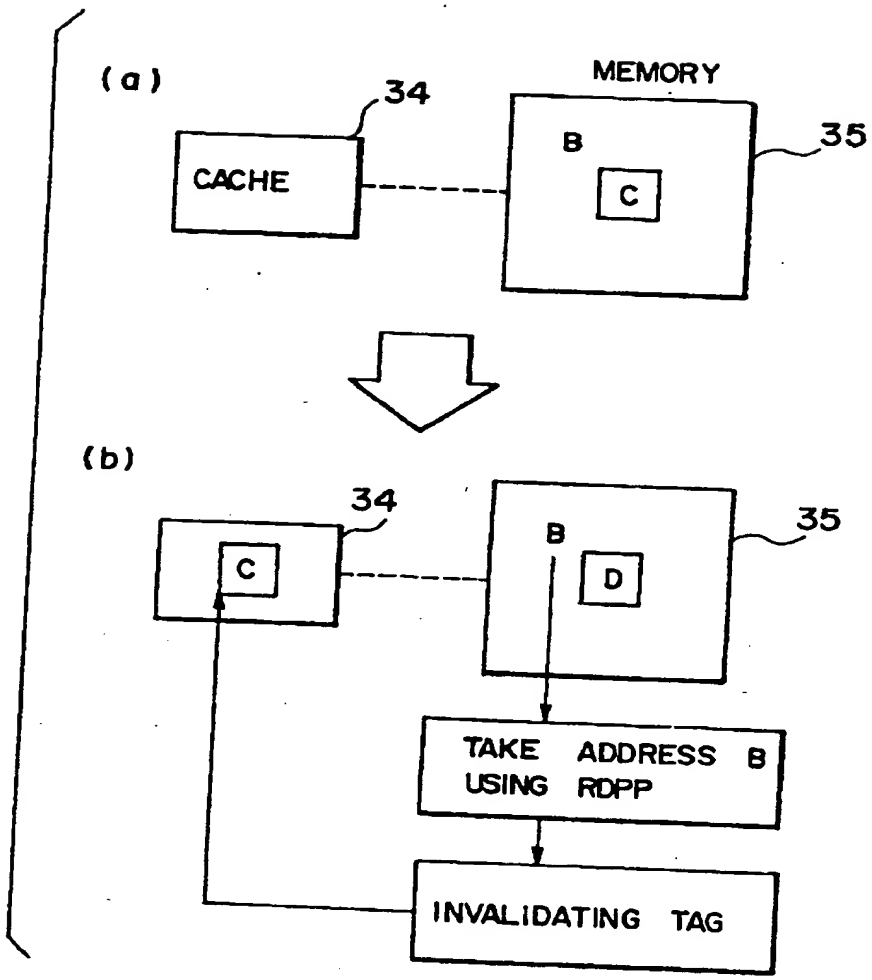

(a)
MEMORY
34
35
B
CACHE
C
(b)
34
35
B
C
D
TAKE ADDRESS B
USING RDPP
INVALIDATING TAG

FIG. 13A

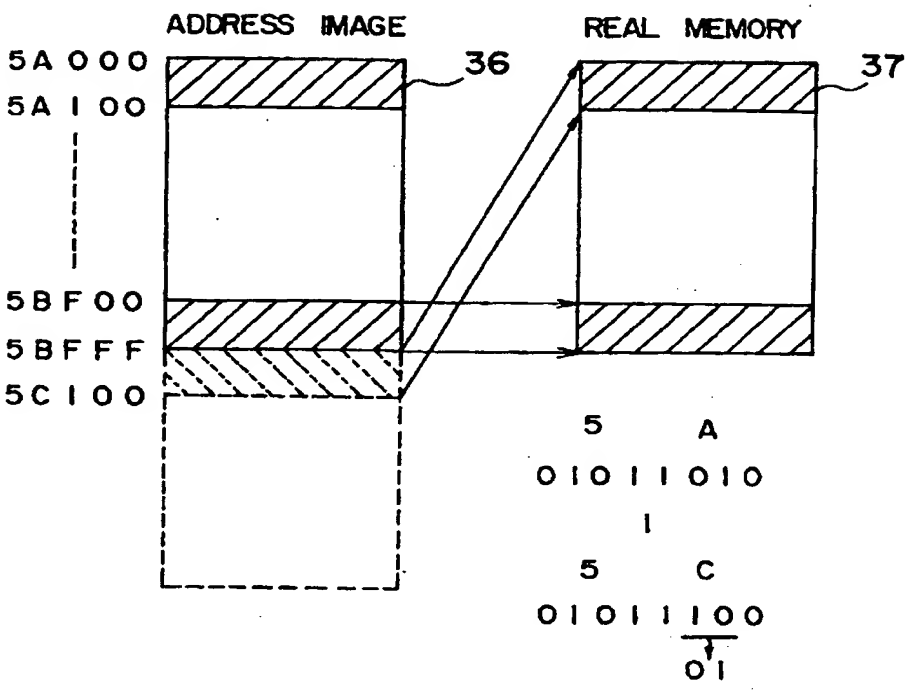


FIG.13B

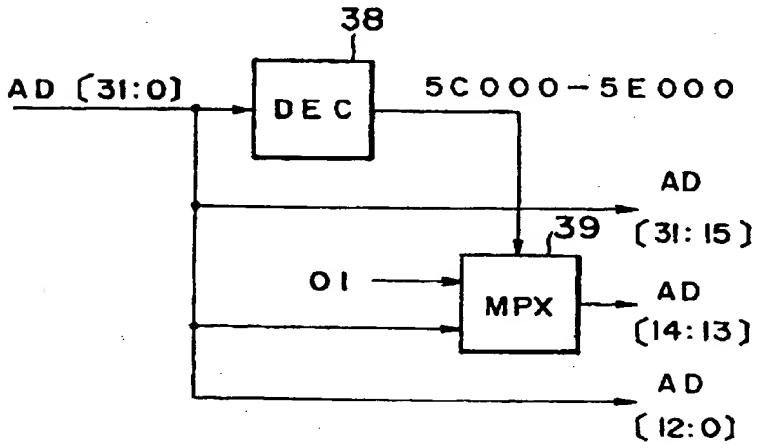
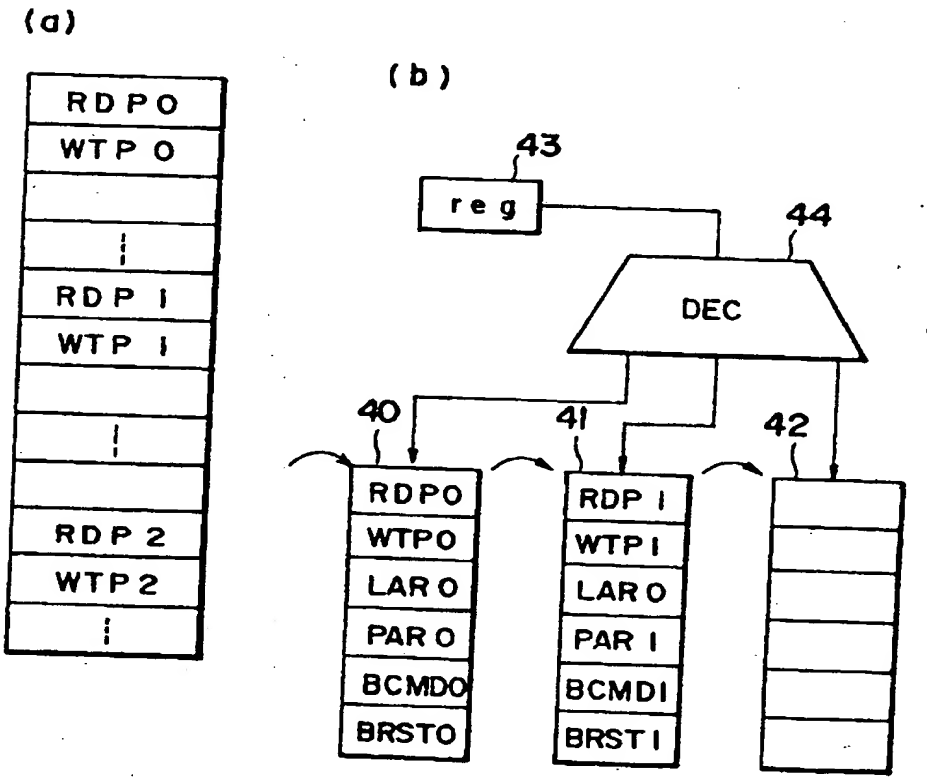


FIG. 14


(a)
RDP0
WTP0
RDP1
WTP1
RDP2
WTP2
(b)
43
reg
44
DEC
40
41
42
RDP0
WTP0
LAR0
PAR0
BCMD0
BRST0
RDP1
WTP1
LAR0
PAR1
BCMD1
BRST1